# Roteiro de Experimento: Watchdog Timer com Reset Automático na BitDogLab

## 1. Título do Experimento

Detecção de Travamentos e Reinicialização do Sistema com Watchdog Timer (Projeto: isr\_wdt\_reset\_bitdoglab)

## 2. Objetivo

- Compreender o funcionamento do Watchdog Timer (WDT) no RP2040.  
- Detectar situações de travamento do sistema utilizando o WDT.  
- Aprender a reiniciar o sistema automaticamente em caso de falha.

## 3. Materiais Necessários

- Placa BitDogLab.  
- LED integrado no GPIO12.  
- Computador com VSCode e Pico SDK configurado.

## 4. Diagrama de Montagem

• LED conectado ao GPIO12.  
• (A BitDogLab já possui o LED integrado, sem necessidade de montagem adicional.)

## 5. Procedimento

### 5.1. Inicialização do projeto

• Criar um novo projeto no VSCode com suporte ao Pico SDK.  
• Nome sugerido: isr\_wdt\_reset\_bitdoglab.

### 5.2. Código Fonte Básico

#include <stdio.h>  
#include "pico/stdlib.h"  
#include "hardware/watchdog.h"  
  
#define LED\_PIN 12  
  
void tarefa1() {  
 static bool recurso\_ocupado = false;  
  
 if (!recurso\_ocupado) {  
 recurso\_ocupado = true;  
 printf(" Tarefa 1 está usando o recurso (e não libera)...\n");  
 gpio\_put(LED\_PIN, 1);  
 while (true) {  
 sleep\_ms(100);  
 }  
 }  
}  
  
void tarefa2() {  
 printf(" Tarefa 2 tentando usar o recurso...\n");  
 gpio\_put(LED\_PIN, 0);  
}  
  
int main() {  
 stdio\_init\_all();  
  
 int tempo\_espera = 0;  
 while (!stdio\_usb\_connected()) {  
 sleep\_ms(100);  
 tempo\_espera += 100;  
 if (tempo\_espera >= 10000) break;  
 }  
 sleep\_ms(500);  
  
 gpio\_init(LED\_PIN);  
 gpio\_set\_dir(LED\_PIN, GPIO\_OUT);  
 gpio\_put(LED\_PIN, 0);  
  
 printf(" Sistema iniciado. Watchdog ativado (tempo = 6s)...\n");  
  
 watchdog\_enable(6000, false);  
  
 while (true) {  
 tarefa1();  
 sleep\_ms(500);  
 watchdog\_update();  
  
 tarefa2();  
 sleep\_ms(500);  
 watchdog\_update();  
 }  
}

## 6. Análises e Observações

- A Tarefa 1 simula um travamento ao entrar em um loop infinito.  
- A Tarefa 2 deveria executar normalmente, mas não é chamada após o travamento.  
- Sem atualização do watchdog, o sistema é reiniciado automaticamente após 6 segundos.  
- Discuta:  
 • Importância do Watchdog em sistemas críticos.  
 • Ajuste adequado do tempo de timeout do Watchdog.

## 7. Extensões e Desafios

- Diminuir o tempo do Watchdog para detectar falhas mais rapidamente.  
- Simular falhas diferentes (não apenas travamento).  
- Implementar salvamento de estado antes de resetar.  
- Usar watchdog\_caused\_reboot() para detectar se houve reinicialização por falha.

## 8. Importante

- A função watchdog\_update() deve ser chamada periodicamente para evitar reset.  
- stdio\_usb\_connected() permite aguardar conexão USB.  
- O Watchdog deve ser configurado com tempo suficiente para distinguir travamentos reais.